2008

Intelligent Spectrum Sensor Radio

Omer Mian
Wright State University

Intelligent Spectrum Sensor Radio

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Engineering

By

Omer Mian
Department of Electrical Engineering
Wright State University

Spring 2008
Wright State University

WRIGHT STATE UNIVERSITY

SCHOOL OF GRADUATE STUDIES

April 11th, 2008

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Omer S. Mian ENTITLED Intelligent Spectrum Sensor Radio BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Engineering.

_____
Zhiqiang Wu, Ph.D.
Thesis Director

_____
Fred Garber, Ph.D.
Department Chair

Committee on
Final Examination

_____
Zhiqiang Wu, Ph.D.

_____
Yong Pei, Ph.D.

_____
Xiaodong Zhang, Ph. D.

_____
Joseph F. Thomas, Jr., Ph.D.
Dean, School of Graduate Studies

ABSTRACT

Mian, Omer. M.S.E, Department of Electrical Engineering, Wright State University, 2008
Intelligent Spectrum Sensor Radio

A cognitive radio is a radio with built-in intelligence that makes it able to utilize the radio frequency spectrum more efficiently by adapting to the changing conditions and frequency availability. In this thesis a spectrum sensor radio is designed that detects transmissions in FM band and determine the range of used and unused frequencies within the band. The radio then starts a transmission on a frequency among the available frequencies. This intelligent radio system design is implemented using the Universal Software Radio Peripheral and GNU Radio with a program written in python that uses a number of GNU Radio modules along with other python and C-Shell scripts giving a working baseline structure of a cognitive radio.

Table of Contents

INTRODUCTION 1

1.1 Cognitive Radios 1

1.2 Software-Defined Radio (SDR) 3
1.2.1 Definition 3
1.2.2 Architecture 4
1.2.3 SDR Open Source Projects 7

GNU RADIO, PYTHON AND THE USRP 8

2.1 GNU Radio – The Software 8
2.1.1 Python 9

2.2 USRP – The Hardware 11
2.2.1 Basic Architecture 12
2.2.2 Daughter Boards 14
2.2.3 Field-Programmable Gate Array (FPGA) 16

INTELLIGENT SPECTRUM SENSOR RADIO 20

3.1 Application Purpose 20
3.1.1 Motivation 20
3.1.2 Objective 22

3.2 Reception Pipeline 22
3.2.1 USRP Front-End 22
3.2.2 Signal Detection and PSD 24
3.2.3 FFT: The Key Tool 26
3.2.4 Sliding FFT Window 28
3.2.5 Spectrum Classification 33

3.3 Transmission Pipeline 38
3.3.1 Data Source 38
3.3.2 Modulation Techniques 39
3.3.3 Data Sink 40

TESTING AND RESULTS 42

4.1 Demonstration Setup 43
4.1.1 Hardware Components 43
4.1.2 Software Tools 47
4.1.3 Testing 49

4.2 Performance Measures 53

4.3 Future Work 53

4.4 Conclusion 54

BIBLIOGRAPHY 56

List of Figures

Figure 1 An Ideal SDR Block Diagram 4

Figure 2 A Pratical SDR Block Diagram 5

Figure 3 Python Script Example 'dialtone.py' 10

Figure 4 USRP Board with 2 Transmitter boards and 2 Receiver boards [4] 12

Figure 5 FPGA for USRP Block Diagram [4] 12

Figure 6 FPGA MUX implementation in Receive Path 17

Figure 7 FPGA DEMUX and AD9862 Chip implementation on USRP [14][4] 18

Figure 8 Function Block Diagram of AD9862 from [15] 18

Figure 9 The RF Spectrum from [17] 21

Figure 10 Averaging the Spectrum Vector 31

Figure 11 Frequency Spectrum of a 4MHz band in FM Band 32

Figure 12 Frequency Spectrum of a the entire FM Band 33

Figure 13 Peak Determination 34

Figure 14 State Diagram for Threshold Determination 36

Figure 15 Using Threshold for Classification of the Frequency Spectrum 37

Figure 16 FM Modulating and Carrier Signal 40

Figure 17 Data Transmit Pipeline 41

Figure 18 The ISSR Setup 46

Figure 19 State Diagram for ISSR Scipts Structure 50

Figure 20 Terminal Window 51

Figure 21 Spectrum Plot 52

List of Tables

Table 1 USRP Daughteboards Available from Ettus Reseach LLC[12] 14

Table 2 Table of Functions 24

Table 3 Developement Tools Needed for GNU Radio 47

Table 4 Libraries Needed for GNU Radio 47

Table 5 OS Version specific SWIG Installations 48

Chapter 1

Introduction

1.1 Cognitive Radios

The term introduced by Joseph Mitola III in his doctoral thesis in 2000, 'Cognitive Radio', has a number of interpretations [1]. However, generally speaking, a cognitive radio is an intelligent radio which is capable of determining its frequencies of action and adapt to the changing frequency usage within that frequency band. In other words, if a radio is capable of setting and configuring its own parameters including "waveform, protocol, transmitting and receiving carrier frequency and networking" autonomously, it can be called a cognitive radio [2]. As the term *cognitive* suggests, the idea of cognitive radio goes beyond just making productive use of unused part of the spectrum but being capable of making human-like

decisions to transmit without obstruction. It can be used for usage pattern recognition as well as establishing network connections with other cognitive devices and radios.

This intelligent radio, in theory, can allow multidimensional recycling in frequency and time and over come bandwidth restrictions; this can eventually evolve a new era in broadband wireless communications which has been having a hard time progressing in the United States. The innovation which makes engineers see Cognitive Radios a possible technology is the Software Defined Radio (SDR) where the "software meets the antenna" [3]. The property of cognitive radio being able to sense the frequencies and transmissions in the environment surrounding it, has made it an unmatched theoretical technology and its basic implementation using GNU Radio and USRP is the subject of this thesis project. GNU Radio is an open source SDR which works closely with the Universal Software Radio Peripheral (USRP, pronounced 'U-Surp'), a hardware designed for GNU Radio [4].

This chapter will introduce the technology which gives, the engineers today, a platform to work towards making this miraculous idea, realizable. The real world factors and current communications advancement obstructions that motivate engineers like me, towards implementing the idea laid out by Joseph Mitola originally, will also be discussed in this chapter followed by the objective of this thesis. The chapter will end with an introduction to the GNU Radio and the USRP which will be discussed in detail in the following chapter.

1.2 Software-Defined Radio (SDR)

1.2.1 Definition

Software-Defined Radio is one of the latest and most evolving technologies in the communications industries in civilian, military as well as commercial sectors [5]. Joseph Mitola defines software-defined radio as a software radio is a radio which is capable of performing channel modulation and demodulation in software. That is, "waveforms are generated as sampled digital signals, converted from digital to analog via a wideband DAC and then possibly upconverted from IF to RF. The receiver, similarly, employs a wideband Analog to Digital Converter (ADC) that captures all of the channels of the software radio node. The receiver then extracts, down converts and demodulates the channel waveform using software on a general purpose processor." [6]

An entirely hardware based radio gives no flexibility because of the fixed characteristics of the modules performing the radio functions. However, in a radio system built with SDR, the flexibility is very high. Basically, software-defined radio is a radio system that can be run on a hardware that is programmable and has "general purpose" hardware that can perform digital signal processing (DSP) algorithms at both the receiving and transmitting side while the radio is in use [2].

1.2.2 Architecture

The idea behind a software-defined radio is to bring the software code "as close to the antenna" as possible according to Eric Blossom (Founder and Architect of GNU Radio). This being said, we can picture an ideal software-defined radio as shown in the figure below:

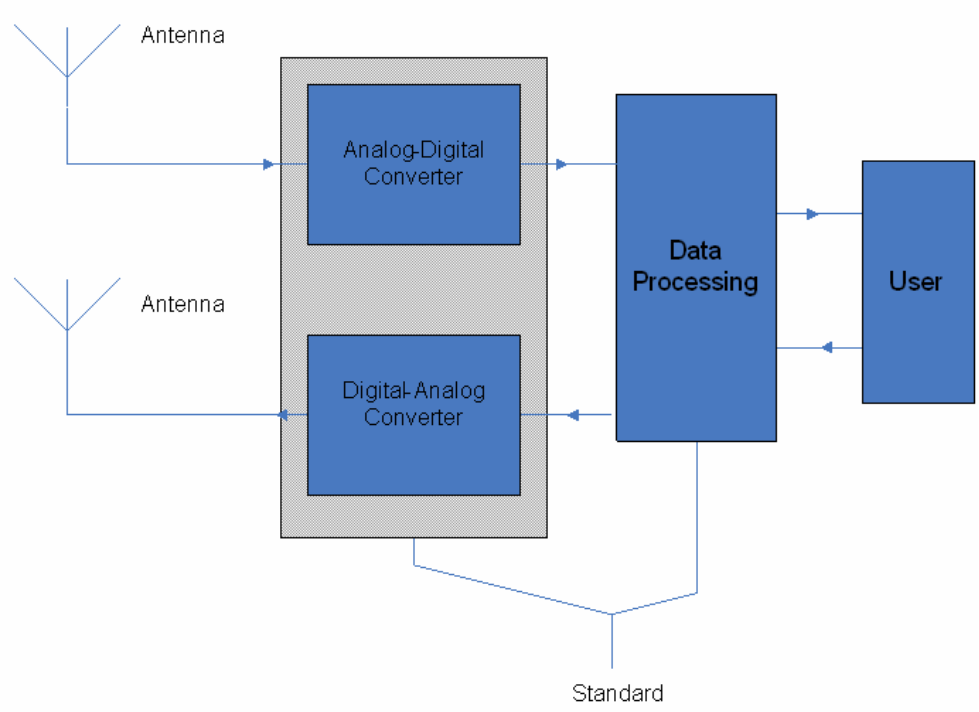

Figure 1 An Ideal SDR Block Diagram

By looking at Figure 1, we see that the idea looks very straight forward. However, in reality there isn't such hardware available that can perform the transmitting and receiving without a RF front-end. Hardware also limits the other aspects of the ideal SDR such as one single antenna cannot receive signals from the entire spectrum but only bands of frequencies and to interpret and demodulate signals at higher frequencies we need extra hardware to make it possible for a computer to properly sample a signal [7].

When the constraints and hardware limitations are considered, it is then seen that practically possible software-defined radio is somewhat different. There are four main sections in a practically realizable software-defined radio: RF section, IF section, baseband section and data section (parameters).

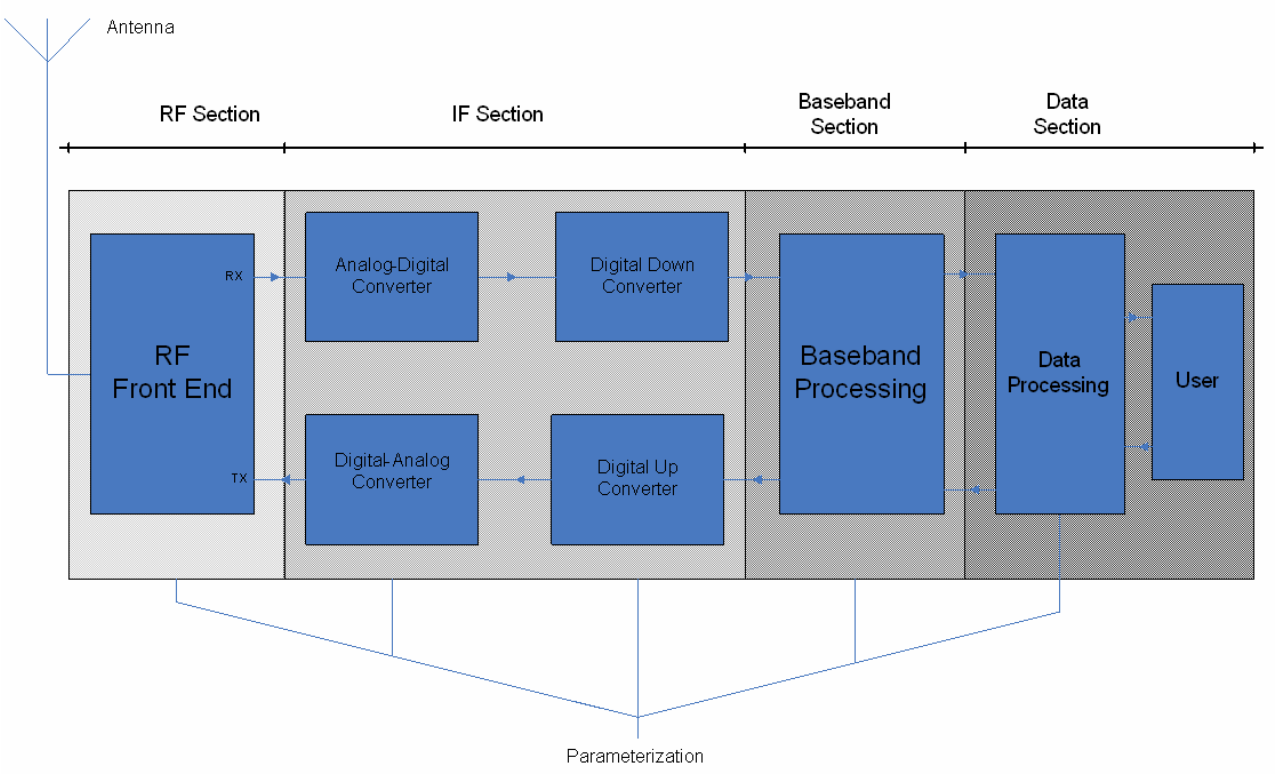

Figure 2 A Pratical SDR Block Diagram

- RF Section: This section contains set of antennas covering the entire spectrum under operation/investigation for both transmission and reception of the signals to the outside frequency world. Contains an RF front-end for tuning, detection, and transmission of the signals. This is also the section mainly responsible of the system's

range and the analog up conversion and down conversion for both transmission and reception respectively to and from IF.

- IF Section: In the IF section is where the most important function of a digital radio, digitizing of the signal and back, takes place. In this section there are both analog-to-digital converter (ADC) and digital-to-analog converter (DAC) on receive and transmit path respectively. This is responsible of making the received analog data computer readable and the computer generated data transferable using RF front-end. In addition to ADC and DAC, digital up-converters (DUC) and digital down-converters (DDC) are used to make high frequency data computable by currently available computer power users have. More importantly, these blocks also perform modulation and demodulation of the signals.

- Baseband Section: This section performs main structure of the operations including the security protocols, correlation etc.

- Data Section: This section sets the parameters for the radio system. It contains the set of instructions and standards that the baseband and the DUC/DDC sections use to perform their functions as they are programmable. This provides the user interface (e.g. PC) to program and develop intelligent controls and other routines for the radio system.

As the technology suggests right now, the programmability of the RF front end is not possible which makes the realization of an ideal SDR impossible also.

1.2.3 SDR Open Source Projects

GNU Radio is an open source software-defined radio project which was started by Eric Blossom intended to support broadcast of digital HDTV. It is a free software and therefore has a complete source code available online and all the functionality and how it is build can be seen easily [8]. GNU Radio is an official GNU project and being free software, is being used in various applications of software-defined radios. This signal processing software tool gives the user a complete package supporting implementation of "oscilloscope, concurrent multi channel receiver and an ever-growing collection of modulators and demodulators" [9].

Along with many other hardware modules including most of the soundcard, tv tuners, and fm tuners that work with gnu radio, the Universal Software Radio Peripheral (USRP) is a hardware made by Ettus Research LLC that is solely made for GNU Radio. It almost performs everything that the GNU Radio is capable of doing and the way it works is going to be discussed in the following chapter along with the working of GNU Radio.

Chapter 2

GNU Radio, Python and the USRP

2.1 GNU Radio – The Software

GNU Radio is an open source toolkit developed by Eric Blossom owner of Blossom Research LLC. Eric Blossom, a person who has knowledge in a wide variety of areas, has done a lot of work in communications.[8] Right now one of his experiments, GNU Radio, is getting the hype from all types communications researchers because of its flexible programmability and low-cost, compatibility with current hardware and free accessibility.

When a software-defined radio is put together, it is noticed from section 1.2 that it contains both hardware and the software that drives the hardware and the peripherals attached to it. GNU Radio is that software in a software-defined radio transmit and receive

structure paths that is accessible to the user and therefore, it is programmable to meet the requirements of the system. GNU Radio is made up of many radio and signal processing blocks which can be grouped together by user in an order to construct waveforms. These signal processing blocks are written in C++ are used as it is without any modification by most of the users. Python (a higher level computer language than C++) is used mainly to give these processing blocks parameters for use and works as a tool box to put all of the blocks together and perform the tasks.

2.1.1 Python

Python is a "dynamic object-oriented programming language" [10] that has proven to have strong integration capabilities with other languages, such as C++. This property makes it versatile and more flexible. A large number of libraries are already available making it even easier for a new person to learn quickly and start working with it in a few days.

Python is a relatively easy-to-learn language with a lot of built-in libraries and data-types built in making it a "very-high-level language". It allows the user to re-use already built modules and also interpret instructions without any compilation or linking. Another important thing is the variable declaration in python. No variable declaration is necessary in Python before it is defined which makes it a lot like Matlab and engineer-friendly.

Since in this thesis implementation, a lot of built-in scripts and functions along with a lot of block-connecting is done, therefore, it would be a good idea to consider the following example, taken from the GNU Radio, of a simple python script that takes advantage of these properties.

```
#!/usr/bin/env python
#
# Copyright 2004,2005,2007 Free Software Foundation, Inc.
#
# This file is part of GNU Radio
#
# GNU Radio is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 3, or (at your option)
# any later version.
#
# GNU Radio is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with GNU Radio; see the file COPYING.  If not, write to
# the Free Software Foundation, Inc., 51 Franklin Street,
# Boston, MA 02110-1301, USA.
#

from gnuradio import gr
from gnuradio import audio
from gnuradio.eng_option import eng_option
from optparse import OptionParser

class my_top_block(gr.top_block):

    def __init__(self):
        gr.top_block.__init__(self)

        parser = OptionParser(option_class=eng_option)
        parser.add_option("-O", "--audio-output", type="string", default="",
                          help="pcm output device name.  E.g., hw:0,0 or /dev/dsp")
        parser.add_option("-r", "--sample-rate", type="eng_float", default=48000,
                          help="set sample rate to RATE (48000)")
        (options, args) = parser.parse_args ()
        if len(args) != 0:
            parser.print_help()
            raise SystemExit, 1

        sample_rate = int(options.sample_rate)
        ampl = 0.1

        src0 = gr.sig_source_f (sample_rate, gr.GR_SIN_WAVE, 350, ampl)
        src1 = gr.sig_source_f (sample_rate, gr.GR_SIN_WAVE, 440, ampl)
        dst = audio.sink (sample_rate, options.audio_output)
        self.connect (src0, (dst, 0))
        self.connect (src1, (dst, 1))


if __name__ == '__main__':
    try:
        my_top_block().run()
    except KeyboardInterrupt:
        pass
```

Figure 3 Python Script Example 'dialtone.py'

The above script generates a dialtone and plays through the soundcard of the computer on a Linux machine. Just like C++ some libraries need to be included in the beginning of the python script. As far as the variable declaration is concerned, if we look at

line 43-44, *ampl* and *sample_rate* are assigned the values and they don't have to be declared before the values are assigned to them just like in Matlab.

Another important property of the way python works is connecting the blocks to create a functional pipeline. On lines 46-47 two of the blocks defined are called *src0* and *src1*. They are given different parameters which in this case is the different frequencies to the signal wave function *gr.sig_source_f()*. This is the 'data source' in this python program and since in all the GNU radio applications, it is a general of the communication pipeline goes from the datasource → datasink with operations in between if needed. In this script we can see the 'data sink' is the soundcard and therefore on line 49-45 the source and the sink is connected with the function called, *connect()*. This gives a general overview of how the basic scripts of python work and most of the scripts created to implement this thesis project are similar including the example scripts from GNU Radio Examples used are similar.

2.2 USRP – The Hardware

Universal Software Radio Peripheral (USRP: pronounced "U-Surp") is a hardware module developed exclusively for use with GNU Radio, by Matt Ettus and his team at the Ettus Research LLC. With microprocessors and digital signal processors on board, USRP became a low-cost resource for current communication researchers that can perform all of the tasks and more of a traditional radio when working together with GNU Radio. As GNU Radio fits right in the basic architecture of a software-defined radio, likewise, the USRP with capabilities of doing a lot of functions in hardware covers most of a software-defined radio's functionality. If we look at the following block diagram we can see it clear how this hardware module together with additional peripherals, can perform some significant operations.

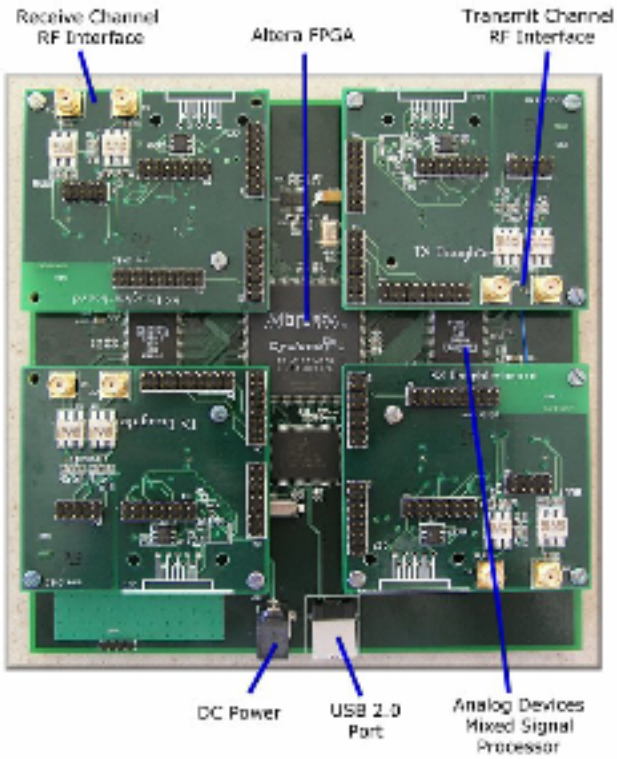

Figure 4 USRP Board with 2 Transmitter boards and 2 Receiver boards [4]

2.2.1 Basic Architecture

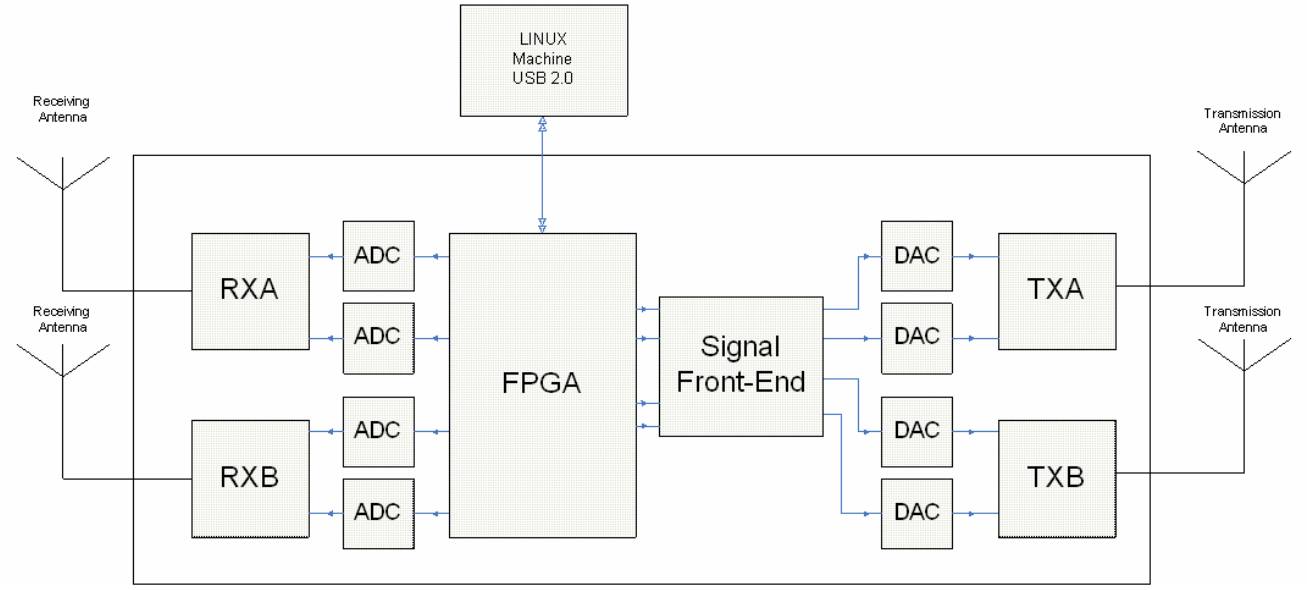

Figure 5 FPGA for USRP Block Diagram [4]

This piece of hardware is the bridge between the real world of analog RF signals and the digital world of the ones and zeros of the computer. The above diagram shows the ADC and DAC working as a bridge between the FPGA (discussed in the next sub-section) and the RF end which in turn brings the digital world closer to the analog world.

The USRP module comes with a motherboard which has USB 2.0 interface for connection to the computer and the power connector. Up to four daughterboards can be connected to one USRP which are clipped on to the motherboard directly most of the time. The four connectors on the USRP motherboard consist of two transmitters, TXA and TXB, and two receivers, RXA and RXB. The USRP currently has four AD Converters with 64MS/s sampling rate and four DA Converters with 128MS/s each with a 2V peak-to-peak with amplified "up to 20dB" with a programmable gain amplifier (PGA)"[4][11]. The USRP gives us 4 inputs and 4 outputs or 2 complex inputs and 2 complex outputs.

Aliasing

As the sample rates are considered for the AD and DA converters, it is obvious that there are some restrictions following the usage of the USRP. Aliasing is an issue that restricts the information a user can get through the process of down conversion to IF and therefore, 32MHz is the highest frequency that can be sampled considering the AD max sampling rate of 64MS/s[1]. Similarly, the DA converter restricts the frequency with which the signal is being transmitted at a maximum of 64MHz because of the 128MS/s max sampling rate. Normally, only 50MHz is used in order to make the filtering procedure simpler[4].

[1] Nyquist Theorem suggests sampling at least twice the frequency of the signal to prevent aliasing.

2.2.2 Daughter Boards

The USRP is capable of using up to 4 daughter boards depending on the daughterboard needed for the specific job. Right now there are several daughterboards available for sale at http://www.Ettus.com. Each of them can be used for different jobs and range for different frequency range, amplification, filtration and tuning capabilities. Following is the list of the daughterboards currently available by Ettus Research LLC at the prices shown blow:

Board	Frequency	Price (USD)
BasicRX	1MHz to 250MHz	$75.00
BasicTX	1MHz to 250MHz	$75.00
LFRX	DC to 30MHz	$75.00
LFTX	DC to 30MHz	$75.00
TVRX	50MHz to 870MHz	$100.00
DBSRX	800M to 2.4GHz	$150.00
RFX900	800MHz to 1000MHz	$250.00
RFX1200	1150 MHz to 1450 MHz	$275.00
RFX1800	1.5GHz to 2.1 GHz	$275.00
RFX2400	2.3GHz to 2.9 GHz	$275.00

Table 1 USRP Daughteboards Available from Ettus Reseach LLC[12]

BasicTX/BasicRX: These boards are pretty much as basic as you can get with the USRP. They contain no filtration or amplification and has no mixer capabilities. An RF front end is not built-in with this daughterboard but can be attached to it [12]. The range of these boards go from 1MHz to 250MHz which means BasicTX can transmit in this range and the BasicRX can receive signals in this range. There is more information including this available in [12]. BasicTX is used in this thesis and will be discussed in detail later on in this chapter.

LFTX/LFRX: LFTX is a low frequency daughterboard which can transmit within the frequency range DC to 30MHz. Similarly the range is the same for receiving for LFRX daughterboard. Both the boards have a low-pass filter of 30MHz.

TVRX: This board is capable of receiving signals only and is used for frequencies ranging from 50MHz to 860MHz. This, noticeably, covers the TV and the FM Radio band and this is why this is a VHF and UHF receiver [12]. The optimum filtration and amplification makes this board ideal for exploring not only TV and FM bands but various other spectrum analyses.

DBSRX: Another receiver among the various daughterboards is the DBSRX board. This covers the frequencies from 800MHz to 2400MHz covering very popular bands of 902-928 MHz ISM, GSM and other key bands [12].

RFX400/RFX900/RFX1800/RFX2400: These daughterboards are transceivers and can perform both receiving and transmitting of signals at a wide range of frequencies. These

three boards together cover a range of 400MHz to 2.9GHz. This can be used for both transmitting and receiving and gives us a low-cost spectrum analysis system through almost the entire range with some added bypassing [12].

TVRX vs. BasicRX:

In this thesis project, both TVRX and BasicTX daughterboards were used to perform the tasks. Two TVRX boards were used for receiving of the signal and the BasicTX for transmitting the intended signal. Initially a BasicRX board was used for receiving the desired frequency band. The BasicRX does not have proper amplification and filtration for the FM Band (88MHz-108MHz) which is the band under analysis in this project and therefore misaligning the FM transmissions with the frequency index. TVRX, although with a higher minimum tune delay[2], not only gives a better reception due to the built-in amplifier but all the tuning and gain (Automated Gain Controllers) tasks are controlled using software [12].

2.2.3 Field-Programmable Gate Array (FPGA)

In order to see and analyze everything that is done inside the USRP or outside in the frequency band needs to be processed by the computer and hence limited to the current computer power available. Therefore, to make this possible there is a FPGA or Field-Programmable Gate Array in the USRP attached to the ADCs and DACs which down converts the high data rate to a level low enough to be transmitted and received through the USB 2.0 (supports upto 480Mbps [13]). This entire process is done by several working

[2] Time for the signal to settle after reception and going through the amplification and filtration

elements within the FPGA which includes DDCs with cascade-integratore-comb (CIC) filters [4] and MUX.

Receive Path: In the receive path of FPGA, the ADCs are connected to the MUX and each IQ of the MUX is connected to a DDC which is connected to a data interleave which puts the data in the receive path queue as showing in the block diagram in Figure 5.

Transmit Path: In the transmit path, the IQ data coming from the queue is CIC filtered and then entered into the DEMUX. The DEMUX is contained within the FPGA however the DACs and the DUCs are outside the FPGA and are in the A9862 chips [4] which are connected to the TXA and TXB as shown in the block diagram in Figure 6.

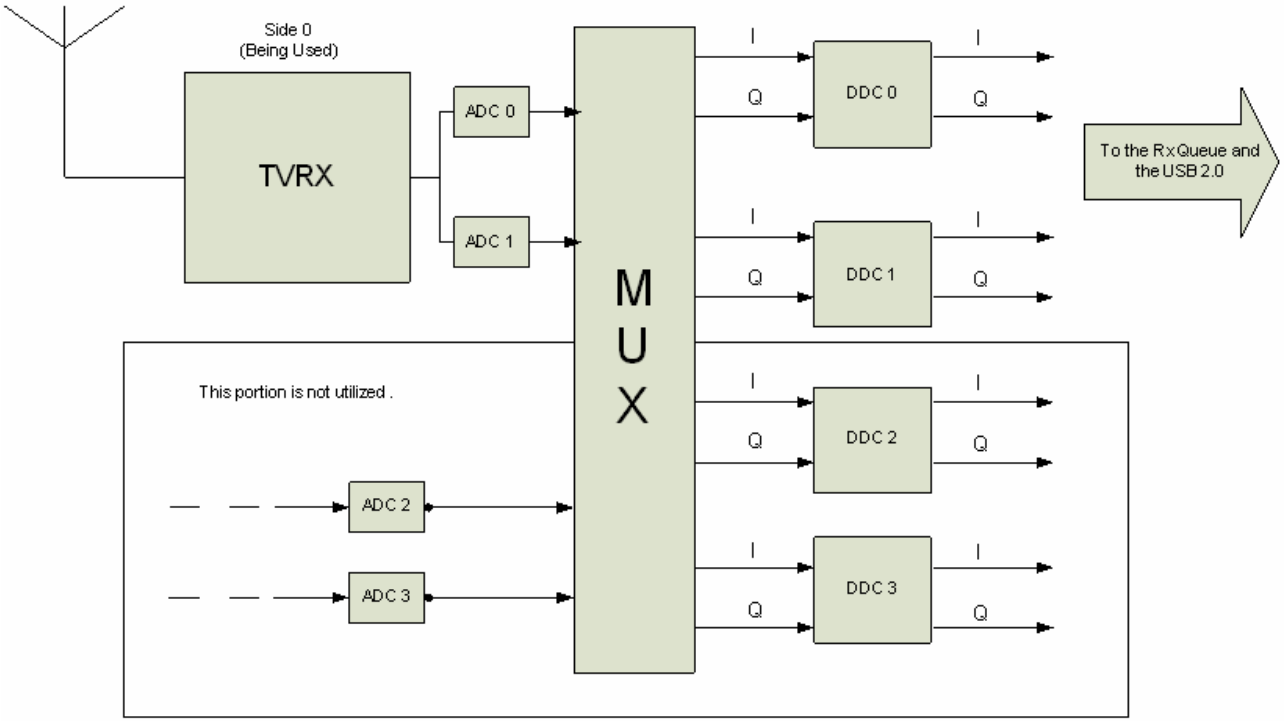

Figure 6 FPGA MUX implementation in Receive Path

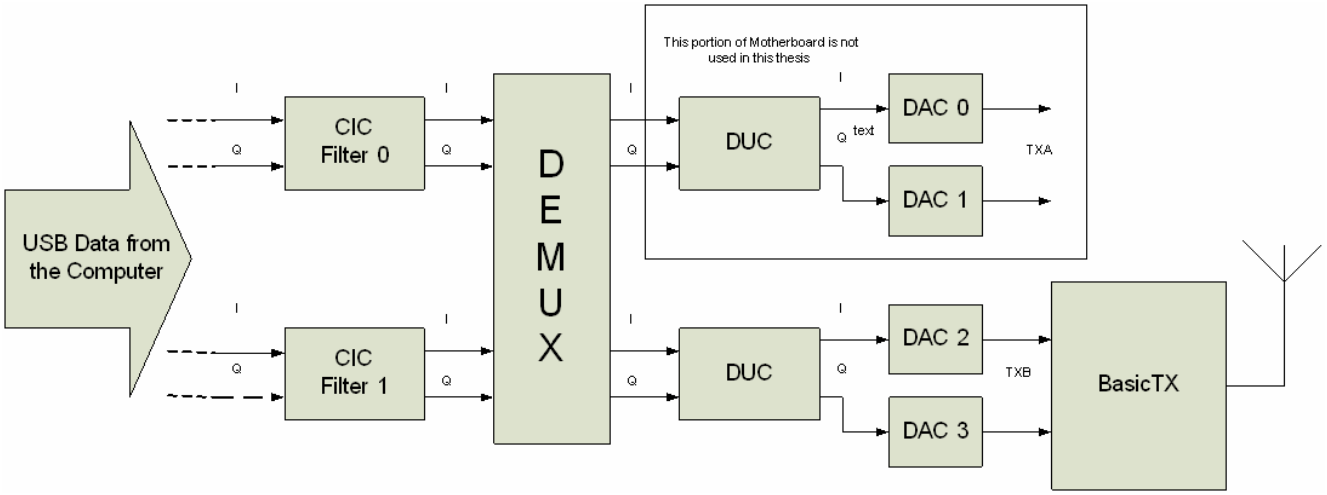

Figure 7 FPGA DEMUX and AD9862 Chip implementation on USRP [14][4]

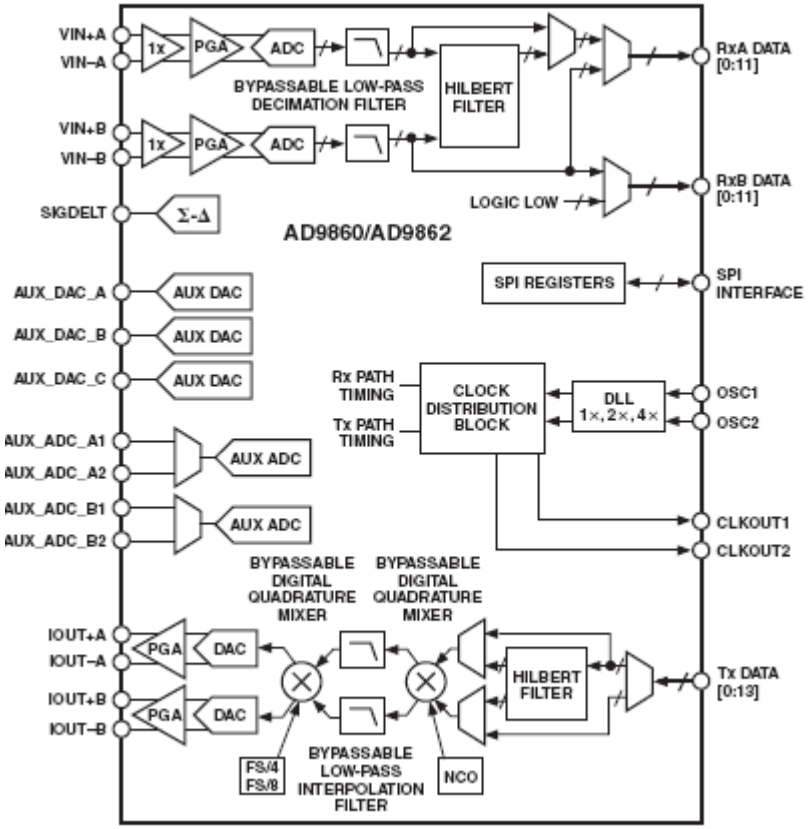

Figure 8 Function Block Diagram of AD9862 from [15]

This thesis is carried out with the GNU Radio software radio and the Universal Software Radio Peripheral. Further detail of the setup will be discussed in the next chapter along with the method used to perform the experiment.

Chapter 3

Intelligent Spectrum Sensor Radio

3.1 Application Purpose

3.1.1 Motivation

RF spectrum, a resource that ranges from DC to 300GHz (limited by current hardware capabilities) is a resource whose value can be imagined when a wireless communication company pays millions of dollars to buy just a tiny bit of spectrum band within that range. This is a limited and a valuable, resource that is protected by the Federal Communications Commission (FCC) approved laws. FCC is responsible of licensed and dedicated use of the spectrum by the current owners of the particular frequencies and bands. By dedicated use, it means that if a company owns a particular band of frequency, it is unlawful and illegal for any other person or company to use that portion of the spectrum for any type of

communication regardless of it being used at that particular time by the owner itself. According to the most of the owners of the band, who have already paid high dollar amounts to own the license to use those frequencies, other communications going on in the band creates high vulnerability risk to their transmissions which can be critical to the business or security depending on the nature of the licensed usage. Since all the licensees were not using the entire spectrum all the time at all places but only a very small percentage in a small region, this creates a huge **surplus** of frequencies that remain unused and hence, saturating the license free spectrum of 2.4-2.48GHz. Following shows the spectrum classifications with respect to the types of usages of the frequencies:

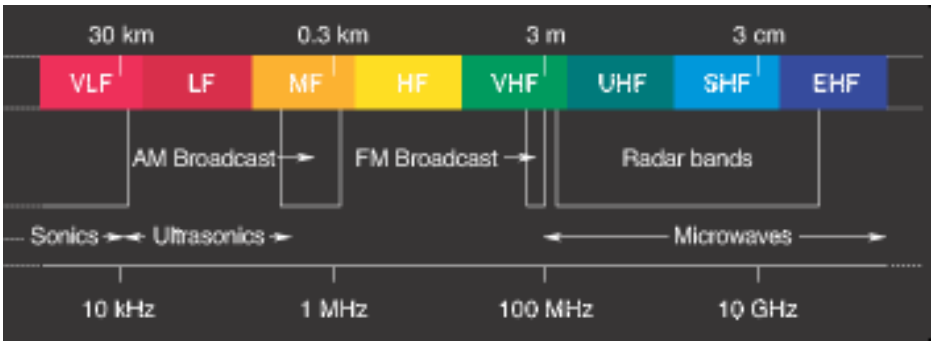

Figure 9 The RF Spectrum from [17]

Cognitive radios that were explained in Section 1.1 provide a solution to the problem of frequencies going idle. These intelligent software based radios can be used for "recycling" the spectrum that goes "wasted". Keeping in mind the frequency spectrum going unused and the idea of cognitive radios, in March 2005, FCC changed the rules saying in a news letter, "In light of the ever increasing demand for radio spectrum, and to facilitate new technologies and services as well as permit more intensive and efficient spectrum use, the Federal Communications Commission today adopted rule changes for cognitive, or "smart," radio systems. This action will facilitate continued growth in the deployment of radio equipment

employing cognitive radio technologies and make possible a full realization of their potential benefits. As a result, consumers will reap the benefit of new and enhanced services." [16]

3.1.2 Objective

Even though the static allocations of the frequency bands have some advantages over the dynamic usage, such as simple communication techniques at both receiver and transmitter, the disadvantage is a lot more severe and serious. This thesis implements the idea of cognitive radio using the GNU Radio and the USRP combination in the FM band (88MHz to 108MHz). The end product is able to sense the entire FM band and perform analysis and estimation routines to identify the ongoing transmission frequencies and spectrum holes (frequencies where no transmission is occurring) for its intended transmission. This proposes a higher spectral efficiency and data rates facilitating an efficient communication protocol. The next section will explain the concepts and methods used to implement this brilliant technique.

3.2 Reception Pipeline

3.2.1 USRP Front-End

Receiving the signal properly and efficiently effects spectrum exploration and could affect the results directly. In this thesis project, mostly the magnitude is the only type of signal information used to for detection and analysis purposes and therefore USRP gives all the information about the signal using some straightforward methods with help of the daughterboards. In this case, the daughterboard used was TVRX which has built-in filters

and amplifier enhancing the signal within UHF band in particular and hence the FM Band making the job much easier.

The properties and the work of common front-end can be guessed fairly well by the information provided about the USRP in Chapter 2. However, this section will focus on the receiver data pipeline in more detail. On the receiving end, USRP is used in combination with one or two daughterboard. Only one daughterboard (TVRX) was used. This combination makes the system able to down-convert a real world analog signal existing in FM Band to an IF frequency signal. This signal is converted to a digital signal using a Mixed Signal Front End Chipset AD9862 which is then decimated down to a 4MS/s in this particular system, so that it can be transferred through a USB 2.0 port to be processed by the computer. AD9862 contains control Programmable Gain Amplifiers (PGA) which is adjusted using feed back parameters [20]. The adjusted stream of data is converted to vector which is, in this case, done using the GNU Radio's built-in *Stream-to-Vector* block. The values in the vector are complex in I and Q format which is converted to magnitudes and used to perform analysis.

The GNU Radio example file *usrp_spectrum_sense.py* was used as the basis performs a lot of these functions already and is part of GNU Radio which is free license software and "can be distributed and modified under the terms of the GNU General Public License" [23]. This script uses the following functions to perform decimation, signal detection, FFT and windowing the details of which are explained in the rest of this chapter:

Function	Operation
source_c	Put incoming data into variable
set_decim_rate	Set decimation rate
stream_to_vector	Puts incoming stream into vectors
fft_vcc	Perform FFT on the data
complex_to_mag_squared	Convert complex vector into magnitude squared
window.blackmanharris	Perform windowing on FFT

Table 2 Table of Functions

3.2.2 Signal Detection and PSD

Accurate and efficient signal detection is necessary for a successful analysis. GNU Radio has built-in signal detection and signal source blocks which most of the time can be used directly to facilitate easy signal detection. The signal detection block is an energy detector that averages the signal over time and estimates the power spectral density (PSD) of the received signal [20].

A power signal has infinite energy and for such a signal Fourier transform does not exist and therefore Energy Spectral Density (ESD) does not exist either. Power Spectral Density (PSD) is an alternative spectral density definition with similar properties as ESD.

If we consider an energy signal *x(t)* which has $0 < E < \infty$ average energy is,

$$E = \int_{-\infty}^{\infty} |x(t)|^2 \, dt$$

Then, for a power signal *x(t)* which has $0 < P < \infty$ average power is,

$$P = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} |x(t)|^2 \, dt$$

As it can be seen, that the average power is basically a windowed version of the average energy of the signal and hence a PSD is a normalized limit of ESD. If ESD of signal $x_T(t)$:

Then, PSD of the signal is:

$$S_x(f) = \lim_{T\to\infty} \frac{1}{2T} |X_T(f)|^2$$

$$P = \int S_x(f) df$$

$$P = \lim_{T\to\infty} \frac{1}{2T} \int |x_T(t)|^2 \, dt$$

In this thesis experiment, the PSD is being worked out by the example file[3] and to get the final estimated PSD the vector coming in from the stream is taken by the computer in sections of fixed lengths, *L*. Digital Signal Processing including FFT and noise minimizing algorithms are performed on this data the details for which is explained in the upcoming section.

There are numerous aspects and techniques involving exploring a spectrum of a particular frequency band and the hardware and techniques available today opens a whole new world to the engineers when they look into the spectrum. One of the techniques is the Fourier Analysis. It assumes that all the real world signals can be approximated as a sum of sinusoids at different frequencies with naturally better approximation when the sum is higher [18].

Fourier Analysis is performed using Fourier transform of the signal which is a "generalization of the complex Fourier series" [26]. Since everything is performed in the

[3] GNU Radio Example file *usrp_spectrum_sense.py*

digital environment, let's look at how the Discrete Fourier Transform (DFT) is defined. Fourier Transform is defined as,

$$F[f(t)] = \int_{-\infty}^{\infty} f(t)e^{-2\pi ivt}\,dt$$

To make it discrete, the argument t can be set to t_n where $n = 0,1,\ldots,N\text{-}1$ making the DFT as,

$$F_n = \sum_{N=0}^{N-1} f_k e^{-2\pi ink/N}$$

A phenomenal application of Fourier analysis is the *frequency spectrum* which is signal harmonic magnitudes plotted against frequencies. This thesis uses the idea of frequency spectrum as part of the spectrum analysis which will be discussed in Section 3.3.3.

3.2.3 FFT: The Key Tool

Nowadays most of the research and development is being done on computers and therefore, to perform Fourier analysis a number of methods were used to perform it on a computer and the technique that is used now pretty much by everyone is the Fast Fourier Transform (FFT) which is a Discrete Fourier Transform (DFT). Going back to the basics, the signal is first transformed to numbers readable by the computer and they are sampled at a fixed rate called *sampling rate*. Computer, therefore, can only work with sample sets instead of continuous signals which has its concerns and tradeoffs of which are:

Aliasing: Sampled signals generate spectrums with only $\frac{N}{2}$ harmonics where N is the number of samples. Therefore, if the original signal has more harmonics than that, it gives errors in the spectrum which is called *aliasing*. This is

suppressed by filtering out the incoming signal for the higher harmonics before the Fourier analysis is carried out on it.

N-points: FFT can only be performed on a sample signal with N samples where N is a power of 2.

Exponential vs. Trigonometric: The FFT is based on exponential series which has a lower magnitude than the trigonometric series.

Spectral Leakage: FFT is performed in a finite interval of time and therefore it is not very frequency selective, hence, resulting in the frequency leakage and low resolution of the spectrum in the frequency domain.

In this thesis implementation python is used and the FFT algorithms in python compensate for this reduction in amplitude. Also, as the methods used to carry out this thesis are discussed, it will be clear that the magnitude values in particular do not play a significant role in the analysis. However, the N-point FFT limitation and spectral leakage is a concern and the way the design deals will be discussed in the later section.

FFT is the key and the most important tool behind the design of the system created in this thesis project. To get information about the FM band and what frequencies are being occupied at a particular time can be determined by looking at the frequency spectrum. A 128-point FFT is calculated of the retrieved signal which gives the magnitudes of the harmonics within the spectrum.

3.2.4 Sliding FFT Window

Blackman-Harris Window:

Spectral leakage problem was discussed in the subsection 3.2.3. This problem changes the result of the FFT performed significantly to make the spectrum analysis extremely difficult and sometimes even impossible. A windowing technique was used to minimize this effect. Windowing is a well established technique to resolve the frequency leakage problem in the spectrum after FFT. This minimizes the higher frequency components or "side-lobes" of the spectrum around the required pulse. In this case, Blackman-Harris window was applied to the signal before the FFT was performed.

Blackman-Harris window "is a straightforward generalization of the Hamming window family". Blackman-Harris window is obtained by adding aliased sinc-function just like any other window family and the definition can be written as following:

$$w_{BH}(n) = w_R(n)\sum_{l=0}^{L-1}\alpha_l \cos(l\Omega_M n)$$

Where $\Omega_M \overset{\Delta}{=} \frac{2\pi}{M}$, and $w_R(n)$ is rectangular window [27]. The Blackman-Harris window transform can also be derived from the rectangular window transform as shown below:

$$W_{BH}(\omega) = \sum_{k=-(L-1)}^{L-1}\alpha_k W_R(\omega + k\Omega_M)$$

Where $W_R(\omega) = M.a\sin c_M(\omega)$ denotes rectangular-window transform and $\Omega_M = \frac{2\pi}{M}$. When L = 3 Blackman family is obtained [27] which is:

$$\varpi_B(n) = \varpi_R(n)[\alpha_0 + \alpha_1 \cos(\Omega_M n) + \alpha_2 \cos(2\Omega_M n)]$$

In this the values if α_0, α_1, and α_2 depends on what kind of degree of freedom is used.

N-Point and Hardware Limitations:

As we discussed in the previous chapter, that the receive path of the USRP is limited to a maximum sampling rate of 64MS/s which means that the computer will see the signal sampled at a decimated value of 64MS/s as the original signal. The decimation value is set to less than half of the 64MS/s which is 16 in this case making the sampling rate $\frac{64}{16} = 4MS/s$. This gives the computer a signal of 4MS/s to work with. As the decimation is made smaller, more and more information in the original signal is ignored but as we go higher, aliasing can occur (if the decimation is higher than 32)[4].

The 128-point FFT is calculated for the incoming signal vector at a center frequency, f_c, to generate the spectrum of the band that shows magnitudes of harmonics of the received frequencies ranging from $f_c - \frac{N}{2}$ to $f_c + \frac{N}{2}$ where N = 128 and hence the resolution of this spectrum. This resolution was considered high enough to get the job done as the frequency band under investigation is FM Band and is well spaced.

Once the FFT was completed and the resultant spectrum vector was plotted, it was not smooth and had a lot of noise despite of the filtration and amplification. This made the harmonic peaks detection difficult. Therefore, *K* numbers of spectrum vectors were retrieved and FFT was performed on them and then they were squared and averaged out giving a relatively smoother curve giving a better picture of the spectrum. The following equation gives a better explanation:

$$\frac{\sum_{i=1}^{K}\left[F\left(s_i(t)\right)\right]^2}{L}$$

[4] Nyquist Theorem suggests sampling at least twice the frequency of the signal to prevent aliasing.

Here the $F(s_i(t))$ is the *i th* vector, which is squared and then added with to the previous vector of sum of all the vectors until *(i-1) th* vector. *L* is the total number of spectrum vectors added to calculate the average.

The following block diagram shows the key steps taken before the threshold value (discussed in Section 3.2.5) is calculated:

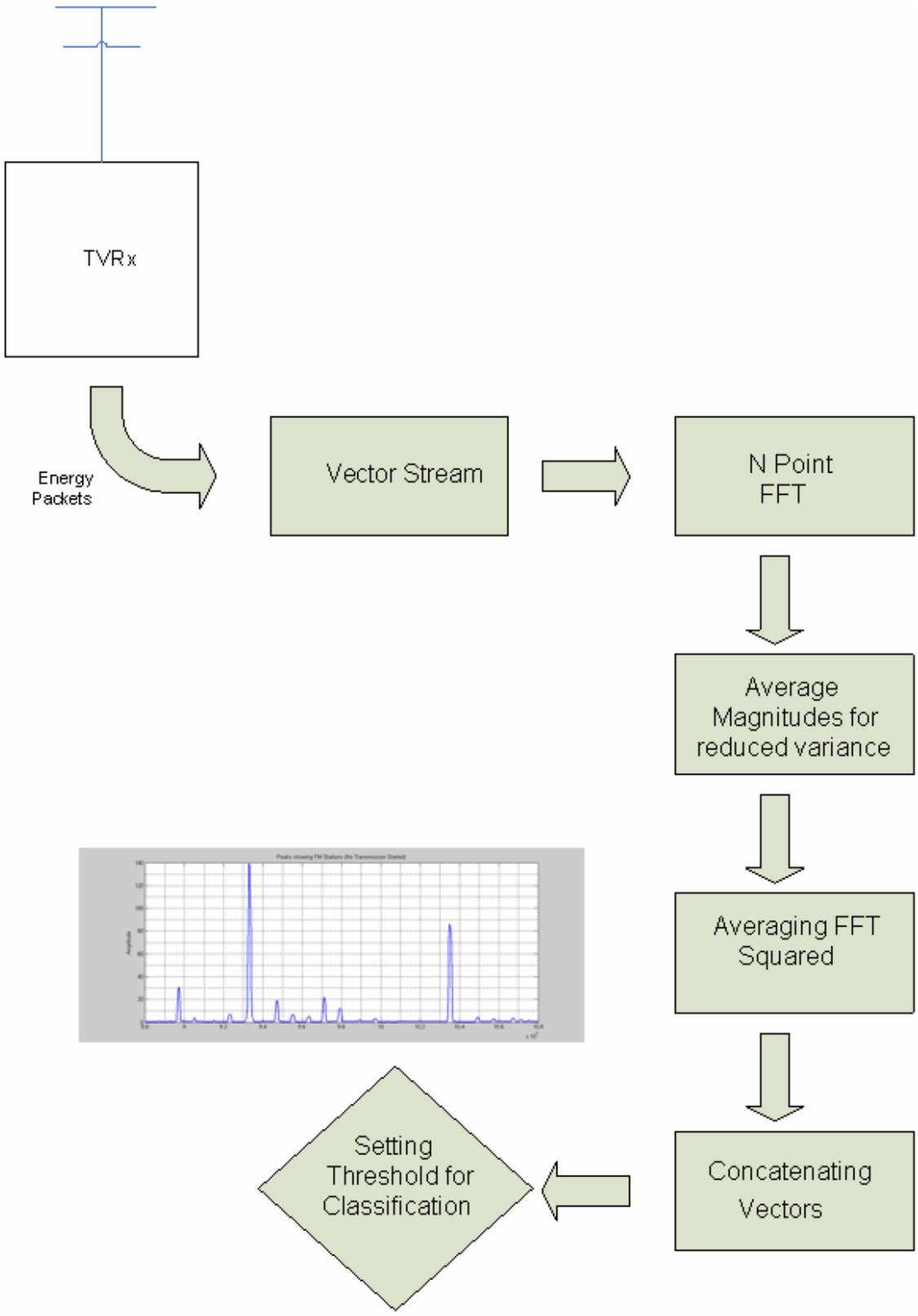

Figure 10 Averaging the Spectrum Vector

This can be seen in the following plot:

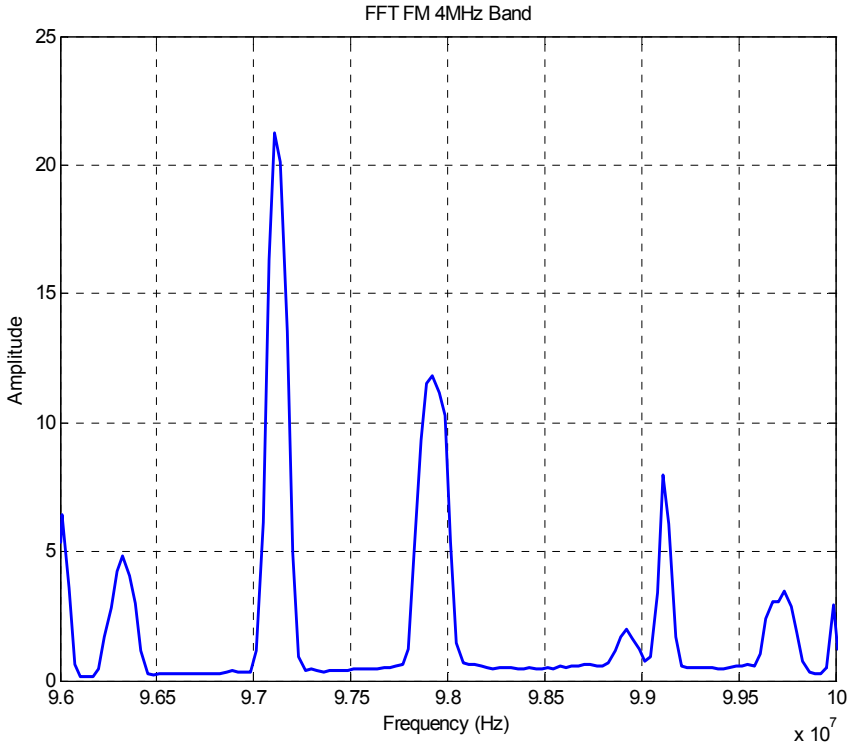

Figure 11 Frequency Spectrum of a 4MHz band in FM Band

It should be noticed that by going this far, the portion of the spectrum that is being covered is 4MHz which is just 20% of the spectrum that is under analysis. Also, going back to Chapter 2, to look at the entire spectrum, idea employed was modifying the *sliding window* of 4MHz at frequencies 4MHz apart in the GNU Radio Example file *usrp_spectrum_sense.py*. FM Band goes from 88MHz to 108MHz and therefore we perform FFT on center frequencies 90MHz, 94MHz, 98MHz, 102MHz and 106MHz covering the entire FM Band. Then concatenating those windows by concatenating the actual vectors, gave the entire spectrum looking like the following figure:

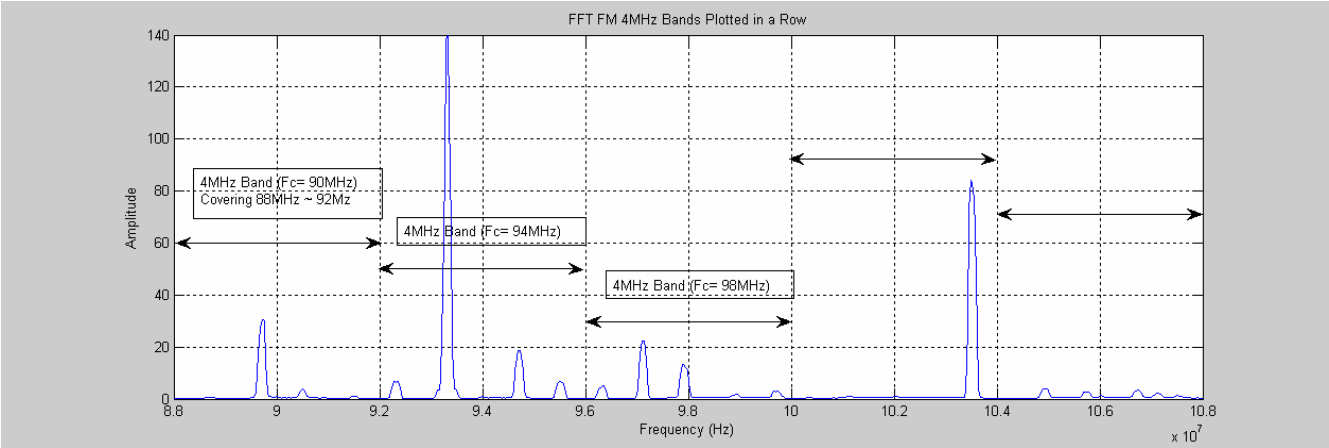

Figure 12 Frequency Spectrum of a the entire FM Band

Now the spectrum for the entire band was able to be worked with decision making and spectrum distribution.

3.2.5 Spectrum Classification

A clean signal with low noise floor is used for decision making if a particular frequency is occupied or not. This process is called *spectrum classification* and is the system's decision making process. In general, when a classification is being done, a criterion should be defined by the system. This gives a threshold or a gap between the classes of the data or spectrum whichever is going through the process. In the design of the ISSR, a threshold value has to be determined, goal being the distribution of the spectrum in two categories. The first is, *Occupied Spectrum* and the second is, *Free Spectrum*. An efficient technique will lead to a more sophisticated spectrum sensing and hence, a more sophisticated ISSR.

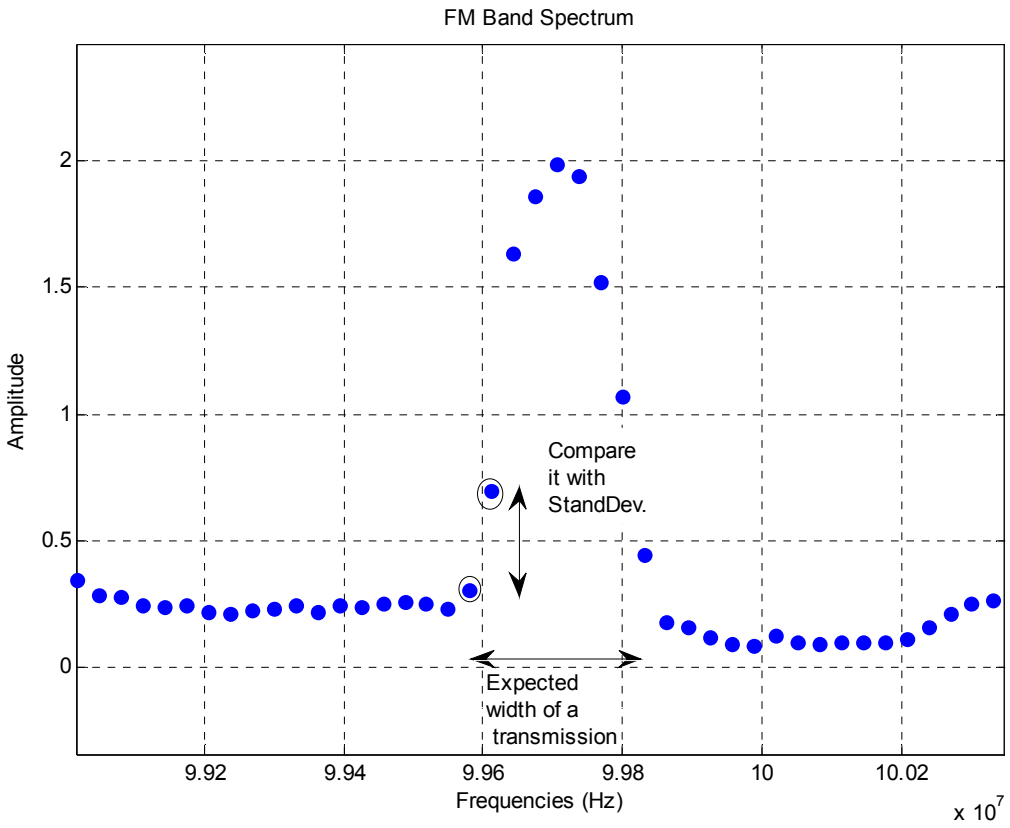

Figure 13 Peak Determination

In order to set a good threshold value, the system goes through the following steps:

1. The spectrum vector magnitude mean, μ_1, is calculated
2. All the magnitudes in the spectrum vector above μ_1 ignored and the mean μ_2 of the resulting vector and standard deviation, σ, of the same is calculated

3. Each magnitude value at frequency f where f_1 is the first value and f_2 is the second value and so on and $f_2 - f_1 = 31.25$ KHz; M_f in the magnitude vector is compared to the value that is 31.25 KHz behind it denoted as $M_{f\text{-}31.25KHz}$.
4. Every time the value of the difference is greater than 3σ, a peak or occupied frequency hole is recognized and total of 8 magnitudes ($M_{f\text{-}31.25KHz}$ plus the next 7 magnitudes) are accumulated (this covers 250KHz and hence a full station) in a variable as a sub-vector. The process starts from the 9th magnitude value again and so on until the entire vector is processed.
5. The maximum values of each sub-vector are compared and the minimum of the maximum values is considered the weakest station detected
6. The threshold is set just above the magnitude value of the weakest station detected in step 5.

Consider the state diagram on the next page for a clearer picture of the algorithm:

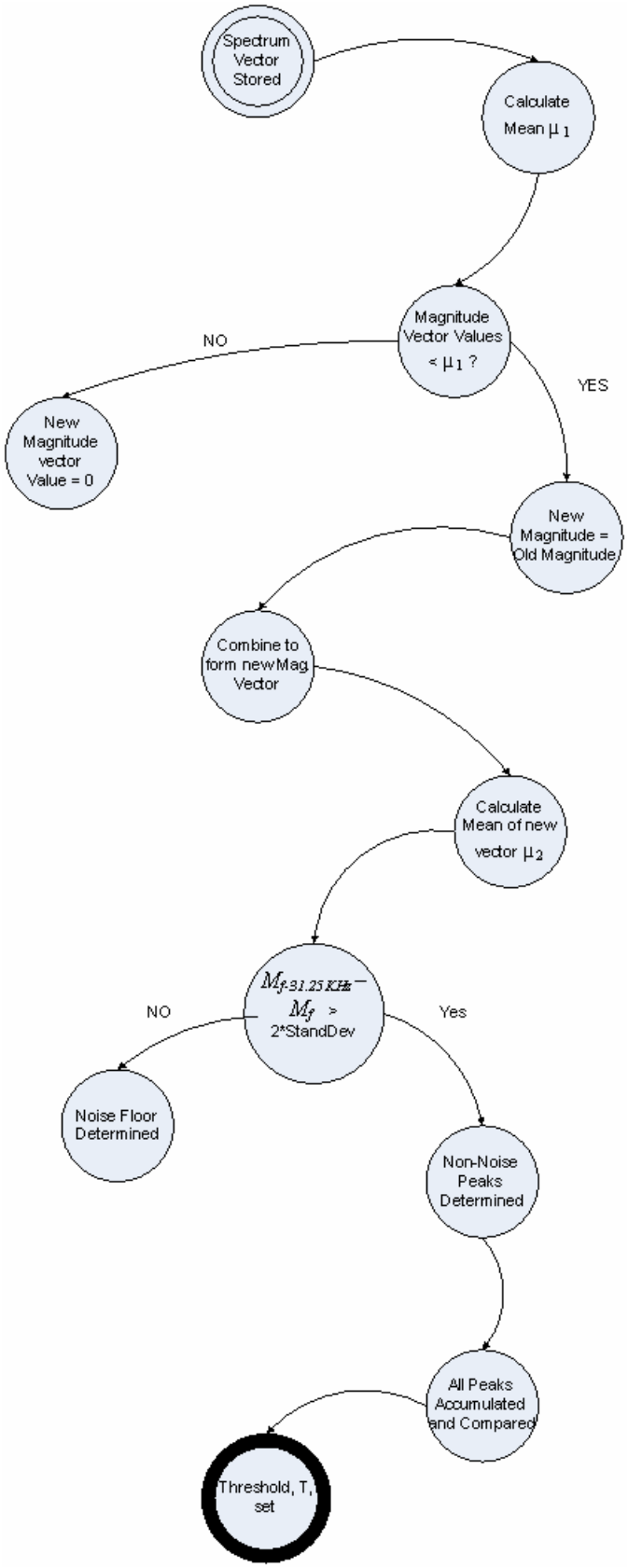

Figure 14 State Diagram for Threshold Determination

The threshold value is the key to a good decision making, and using the above algorithm the value can be tested very easily by just looking at the spectrum plot. The testing will be discussed in detail in Chapter 4. However, once a threshold value *T* is determined the decision, if the spectrum is free or not, is performed based on the following criterion:

Free Spectrum < T

Occupied Spectrum > T

This simple technique is applied to the entire vector and the *Free Spectrum* and the *Occupied Spectrum* is determined. The following picture gives a good idea how this process works:

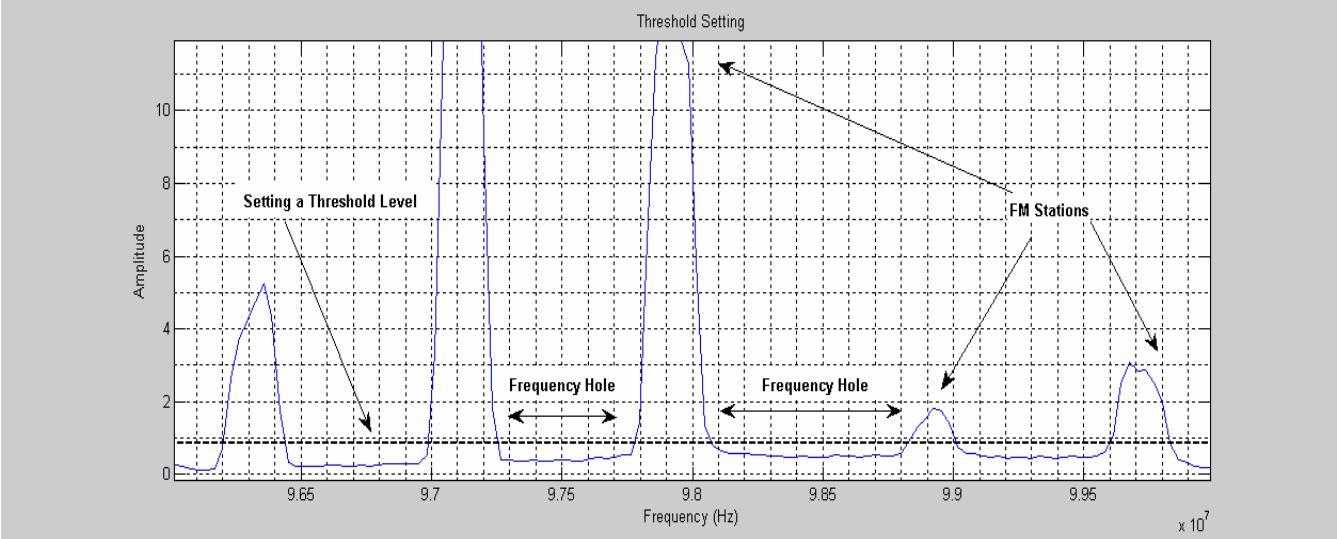

Figure 15 Using Threshold for Classification of the Frequency Spectrum

Looking at Figure 15, each grid-block along the x-axis is 100 KHz wide. From what is in the picture, in the first grid-block the spectrum is below the threshold 100% of the time and therefore it is considered a free frequency and so is the next grid block and therefore in the above particular section the frequency, 96.0MHz to 96.2MHz is considered a spectrum hole

that can be used for transmission. The third and the fourth grid-block along the x-axis show spectrum above the threshold for almost over 75% and 100% of the block respectively giving an occupied spectrum range from 96.3MHz to 96.4MHz. This process goes on until all the 100 KHz blocks are categorized throughout the FM Band calculating all the holes. A frequency among the frequency holes is chosen by the ISSR and the transmission is initiated. The next section will provide details about the transmission protocol.

3.3 Transmission Pipeline

Understanding the transmission protocols that the USRP and the GNU Radio have, when used in combination, is very important before a signal can be transmitted with the correct properties. The powerful digital signal processing platform that USRP and GNU Radio provide makes it possible to transmit a data in various different formats and air interfaces. How the USRP hardware works in conjunction with GNU Radio is discussed in Chapter 2. In this section, we focus on the transmission roadway that the data uses to get from the source to the antenna.

3.3.1 Data Source

Generating digital signals is essential to create or aid a data source to get transmitted. GNU Radio and USRP provide various fast programmable and digital techniques that make digital signal generation easily possible. Modulation, filtration and other digital signal processing techniques are very important for a radio system which can be done by generating “synthesis waveforms” which can be done very efficiently in a digital environment [21].

This thesis dealt with a bit-stream input. Just like a software-defined radio should, GNU radio and USRP contains processing blocks for bit-stream that are programmable and inducible with varying or static parameters as the environment and standards change [22].

The data source was chosen to be the audio card mostly during the experiment. The details of the use will be discussed in the later chapter. The generated IQ values coming from the data stream via USB 2.0. This goes through the CIC filters before getting fed to the multiplexer. An important step is to sample the signal properly and setting the DAC rates so that the data can be transmitted without any over/under run. Before that is performed, the data is up-converted by passing thought the digital up-converters (DUC) and fed to the DACs which then is ready to be transmitted using the transmitter daughterboard. This was performed by the BasicTX in this case.

3.3.2 Modulation Techniques

Signal has to be modulated using a technique that is known to the receiver and is able to be demodulated at the receiver side. Several modulation techniques came across in the thought process of designing the transmit path. Since the band under investigation was FM Band, the frequency modulation technique was implemented and application details of which will be discussed in the next chapter. Using the BasicTX a FM Signal was intended to be transmitted at a particular free frequency determined by the system.

Frequency modulation is an analog technique of modulating the signal using varying frequency of the carrier signal corresponding to the amplitude of the modulated signal. The following figure from [22] shows a typical FM Signal with the modulating signal:

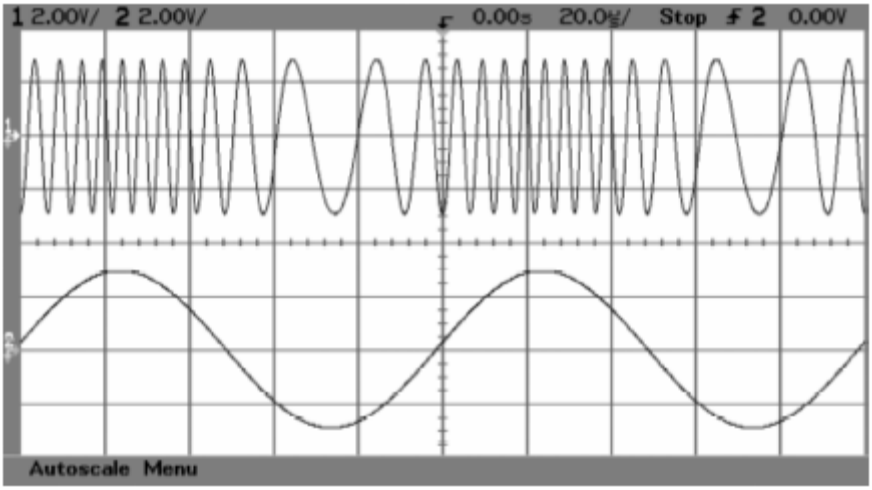

Figure 16 FM Modulating and Carrier Signal

FM Band:

FM band is a band dedicated for being used as broadcasting FM radio stations. “FM” in the FM Band stands for Frequency Modulation which can be confused with the idea that frequency modulation technique can only be used within this band how ever it can be used within a much longer range (VHF/UHF) [19].

The Intelligent Spectrum-Sensor designed in this thesis project was made keeping in mind that the FM stations are approximately 200KHz wide with up to 100KHz wide audio frequencies, with maximum deviation of ±75KHz [19] and therefore the resolution is set to 100KHz for receiving and transmission.

3.3.3 Data Sink

According to the way the GNU Radio blocks interact with the USRP, the blocks of GNU Radio need to be able to t completed the entire path from the source to the sink. The data sink usually being the USRP unless it is not being used to transmit. In the transmission line, the following steps complete a full working transmission path:

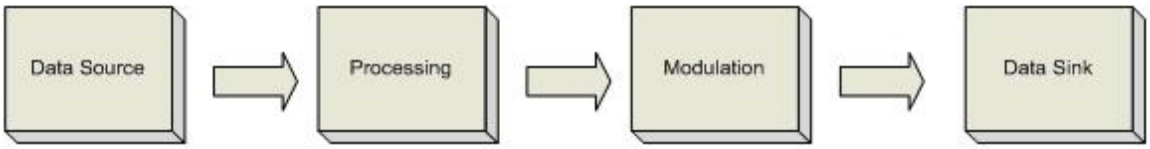

Figure 17 Data Transmit Pipeline

Keeping in mind, a hardware configuration was set up to carryout the design explained in this chapter. The working of the set up and its results are discussed in detail in Chapter 4.

Chapter 4

Testing and Results

It was essential to implement the designed ISSR and therefore a strategy was outlined in order to test the working of system. The objective was to have a system analyze the FM Band and determine frequency peaks and holes in the band regardless of the type of communication being carried out. Once the holes are determined, the system should be able to transmit an MP3 song over the air using the FM modulation. This MP3 song could be a song on the computer or an external device (e.g. iPod, MP3 player, microphone etc.). This should be able to be picked up at that frequency using a FM radio or another USRP. This chapter talks about the experimental setup that was used to perform the given tasks.

4.1 Demonstration Setup

4.1.1 Hardware Components

It is important to get the compatible hardware components for the set up and therefore a careful consideration is given to this part of the thesis experiment. Working with GNU Radio and the USRP a number of things were taken into consideration including the operating system of the computer, soundcard on the computer, the USB interface version and so on before even system building began.

The Computer:

According to [4], GNU Radio at the time of writing, worked best with on a Linux machine and a lot of support available online. In particular it was said, that the GNU Radio works best with Fedora however, it was later found out by trials and some experimenting that all the hardware and software tools needed for the set up ended up working flawlessly with Ubuntu 7.04 Feisty Fawn. The compatibility of GNU Radio and the USRP being universally well known and well tested was not a big concern. The important step was choosing the right Linux machine and in this case the thesis experiment was performed on a Dell Dimension 8300 a dual boot Windows XP and Linux machine using Ubuntu 7.04 Feisty Fawn. The detail specifications of the computer used are as follows:

Make and Model: Dell Dimension 8300

System OS: Dual Boot Windows XP and Ubuntu 7.04

Processor: Pentium 4 3.0GHz with HT Technology

System Memory: 1024MB

Sound Card: Sound Blaster Audigy 2HX

Graphics Card: ATI 9600XT 128MB DDR SDRAM

USB Interface: USB 2.0

System Storage: 400GB

Network Card: Belkin Wireless Adapter

This computer fulfilled the required capability parameters, such as the USB 2.0 Interface, a compatible soundcard, the correct operating system along with a good processing speed.

USRP and the Antennas:

The Universal Software Radio Peripheral motherboard and the daughterboards have already been discussed in detail earlier in this report. In order to fit the needs of this experiment, the USRP was used in conjunction with two daughterboards. These were the TVRX and the BasicTX. The details for both of these boards can be found in Chapter 2 Section 2.2.2.

The TVRX was the main receiving front-end for the system and the BasicTX was used as the main transmitting end. As the band under investigation was FM Band, the receiver card was chosen to be the TVRX because of its built-in filtration and amplification optimized for the UHF band which includes the FM Band. This daughterboard was clipped directly on the USRP, on the RX B socket. The choice between RX A and RX B socket was totally random. To receive FM signals a coaxial antenna was used which was attached to the TVRX board directly. This antenna was approximately 1 meter long and came with a commercial FM radio. The gain for this antenna was unknown and was entirely controlled using the python program itself.

The BasicTX was used to transmit the signals over the air in the FM Band. It was used as it is without adding any filters or amplifiers to the hardware. The gain and amplification was controlled entirely using the software. The daughterboard was directly clipped on the TX A socket which was again a random choice and didn't affect the experiment in any way. Firstly, an FM loop antenna was used to transmit signals however it was experiencing a high attenuation even in a fairly calm and un-obstructive environment. Therefore, a stronger laboratory antenna was used the gain of which was again unknown.

The Audio Source:

Two audio sources were experimented in this thesis which was equally successful. The first was an MP3 song which was on the computer. The song choice was random and didn't have to be specific as far as the song type itself, the sampling rate or the bit rate. The software tools, discussed in the Section 4.2.2, were capable to transform the song very easily to a sample rate and bit rate compatible with the USRP. The USRP performs the DSP and uses the sampling of 32KS/s after interpolation from DAC rate of 128MS/s and the same sampling rate was used for the songs. The song was converted to raw format and using the USRP's *file_source()* block, it was set to be the data source.

The second source that was also experimented was the sound card of the computer. A microphone was attached to the input of the soundcard of the computer. This was the analog audio source converted to digital signals by ADC (also the soundcard). The sampling rate of the soundcard was however fixed at 48KS/s meaning that the sampling rate of the USRP was to be brought close to this if not exactly the same. Therefore the USRP interpolation rate was

set to 320 and software interpolation rate was set to 8, setting the sampling rate to 128MS/320/8 = 50KS/s. This causes a tiny bit of data overrun but not noticeable enough when the transmission is heard on a FM radio.

The Setup:

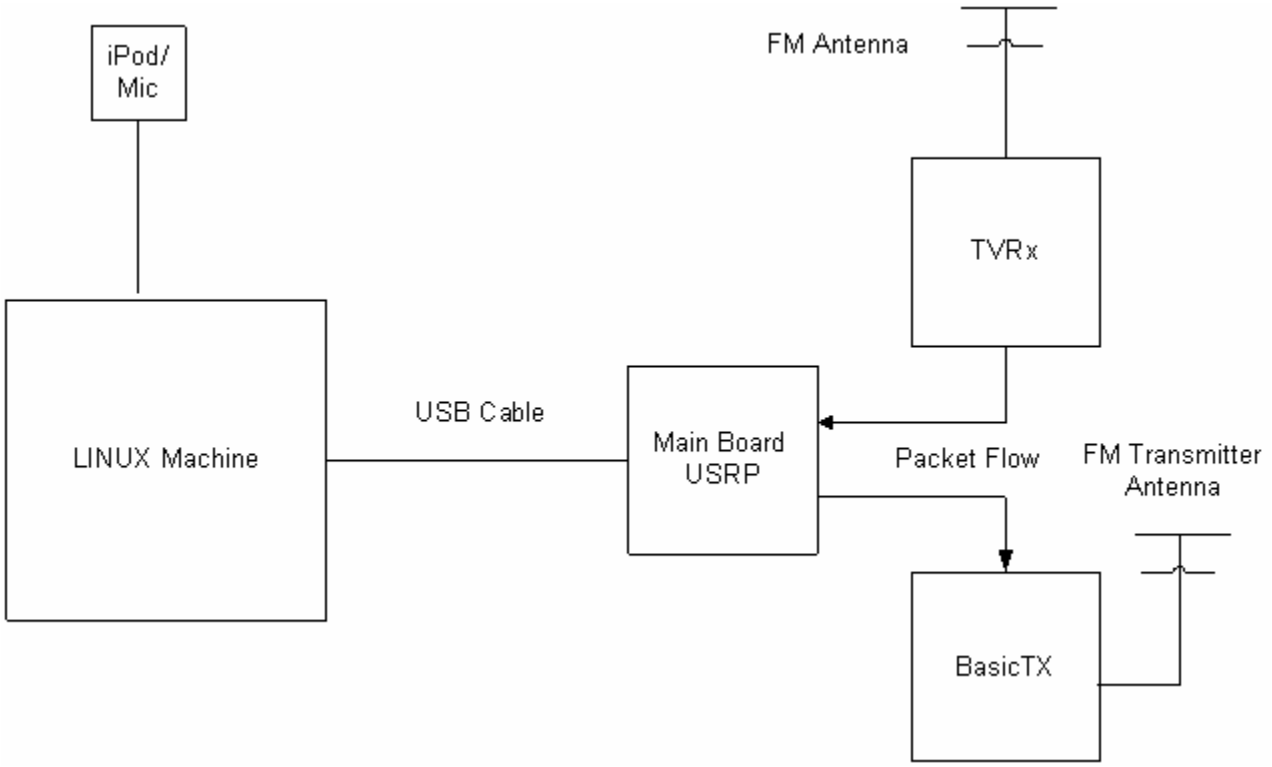

Figure 18 The ISSR Setup

The ISSR was set up as shown in Figure 18. It can be seen that the set up is fairly simple and easy to hook up. As important as it was to get the hardware set up as shown, software tools were also critical.

4.1.2 Software Tools

GNU Radio:

The main software to drive the entire experiment was the GNU Radio. Its proper installation on the computer was critical along with all the blocks. The instructions from [24] were closely followed to install the prerequisites and the GNU Radio itself. The version of GNU Radio that was used is 3.0.3. Specific instructions for installing GNU Radio on Ubuntu system was followed as provided in [24] and a list of libraries and packages required in addition to GNU Radio are as follows:

This list was directly taken from [24].

Development Tools (need for compilation)
g++
subversion
make
autoconf, automake, libtool
sdcc (from "universe"; 2.4 or newer)
guile (1.6 or newer)

Table 3 Developement Tools Needed for GNU Radio

Libraries (need for runtime and for compilation)
python-dev
FFTW 3.X (fftw3, fftw3-dev)
cppunit (libcppunit and libcppunit-dev)
Boost (libboost and libboost-dev)
libusb and libusb-dev
wxWidgets (wx-common) and wxPython (python-wxgtk2.6)
python-numpy (via python-numpy-ext) (for SVN on or after 2007-May-28)
ALSA (alsa-base, libasound2 and libasound2-dev)
Qt (libqt3-mt-dev; version 4 does not seem to work)
SDL (libsdl-dev)

Table 4 Libraries Needed for GNU Radio

SWIG (1.3.31 or newer required)
Edgy or previous: requires installation from source
Feisty or newer: use the standard package install (swig)

Table 5 OS Version specific SWIG Installations

The successful installation was performed using the instructions provided on the GNU Radio website.

Python v2.5:

Among the prerequisites of GNU Radio the most important was the python. This is the compiler and the library required for python programming which is the main driver of the the GNU Radio and the USRP. This high-level language talks directly with the USRP hardware and was used to write scripts to perform the required work for this thesis project. All the example and pre-built blocks to use with GNU Radio are written in C++ and Python. Usually C++ blocks are not required to be modified and most of the programming is done in python.

Matplotlib:

This is a library that was installed in addition to the required library. This software gives python an ability to generate plots and tables for analysis without using the Python GUI. It also gives python a MATLAB-Like interface and therefore a lot of tasks can be easily performed that takes much deeper knowledge of python to be done.

SOX:

SOX is an audio editing software which is extensively used on Linux machine to play, edit, and switch formats of audio files from one to another. This command line operated software. This software is able to work with both types of files including audio files with header information, and the raw (header-less audio data). The key information needed most of the

time by the SOX program is the sampling rate of the output file, channels, data encoding and the amount of data to be converted [25].

C-Shell:

C-Shell is a programming language that was used to link different python scripts for execution purposes of this thesis program. This program gives the user C-language style syntax to run commands on a Linux based system.

4.1.3 Testing

Having all the necessary software tools and the hardware configuration, the experiment was carried out. When the entire hardware is connected, the terminal window directory is changed to the directory with all the required files which had to be put in one directory. It is easier to look at a state diagram, in Figure 19, of all the python and C-Shell scripts to see how they were called, before looking at the actual screenshots of the experiment's proceedings.

From the diagram below, it can be seen that the system reassess the environment N number of time while $N < n$. This is a preset value and can be set to any integer number to tell the system how many times it should reassess. This can also be determined by the number of sections the transmitted data is divided into or the number of times it is to be sent. If another transmission is initiated at the frequency currently being used, the next time the system re-assess the environment, it will avoid that frequency and start the transmission on the next available frequency.

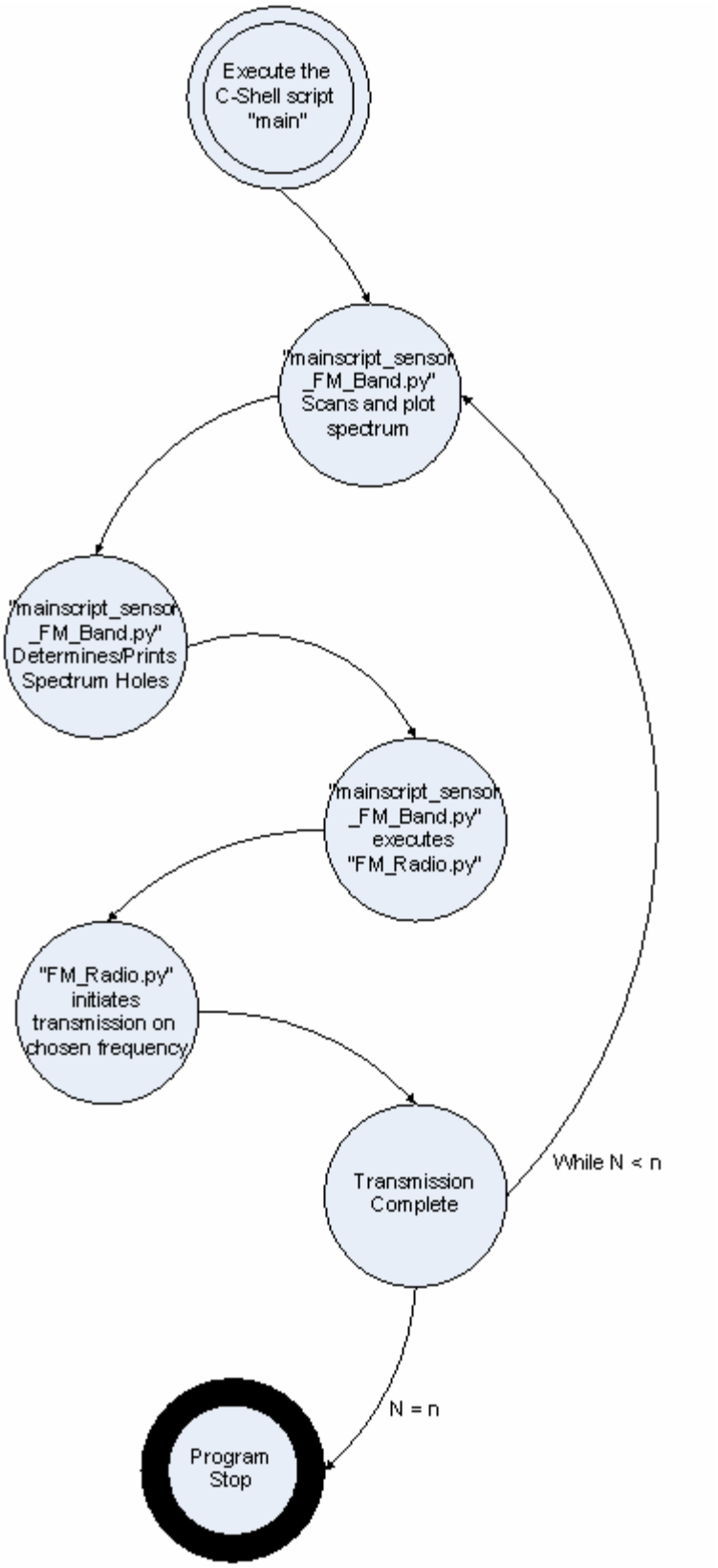

Figure 19 State Diagram for ISSR Scipts Structure

As the program is used and the “main” script is run in the Linux terminal, the following screenshots show what the user will see on the screen.

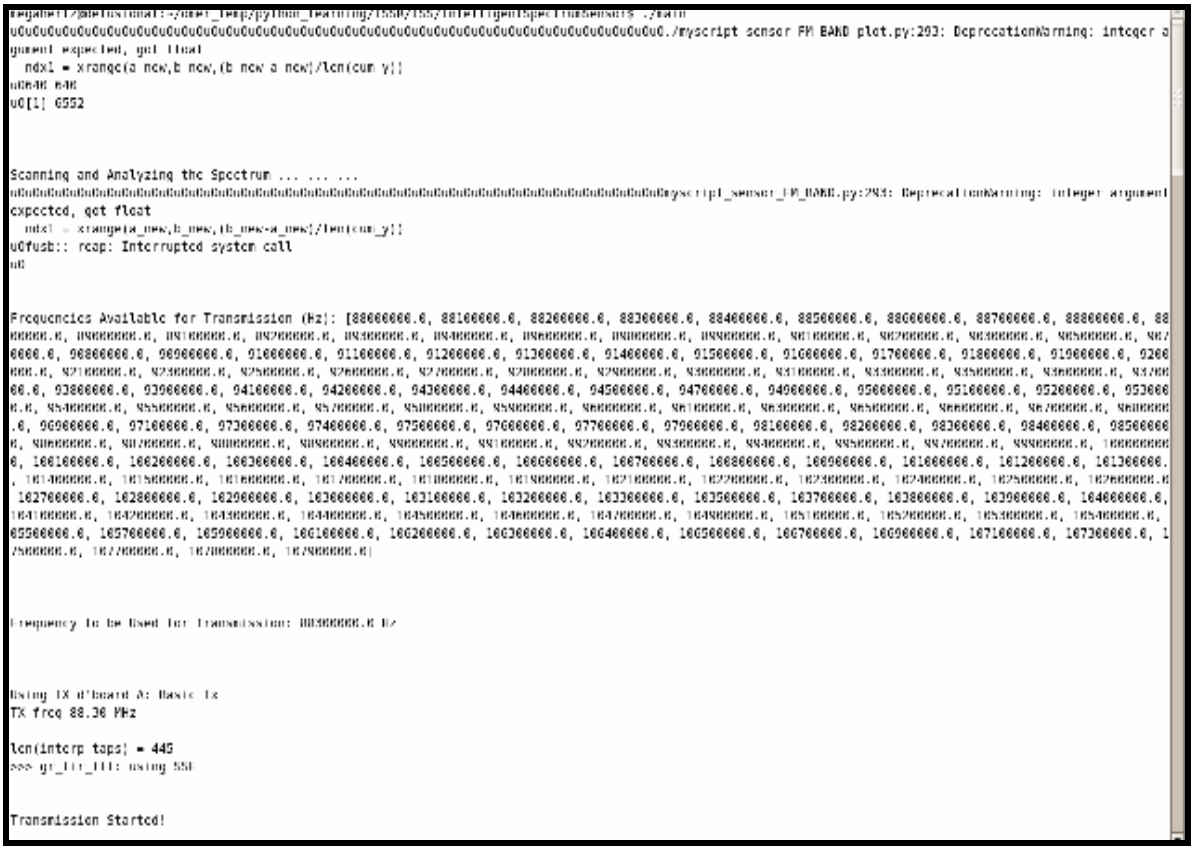

Figure 20 Terminal Window

The above screenshot of the window shows as the program is run, the spectrum is analyzed and the free frequencies are printed out. This screen also shows the chosen frequency to be 88.3MHz in this particular run for the intended transmission. After the analyzing of the spectrum is completed, a snapshot of the spectrum is plotted also which can be seen in the Figure 21.

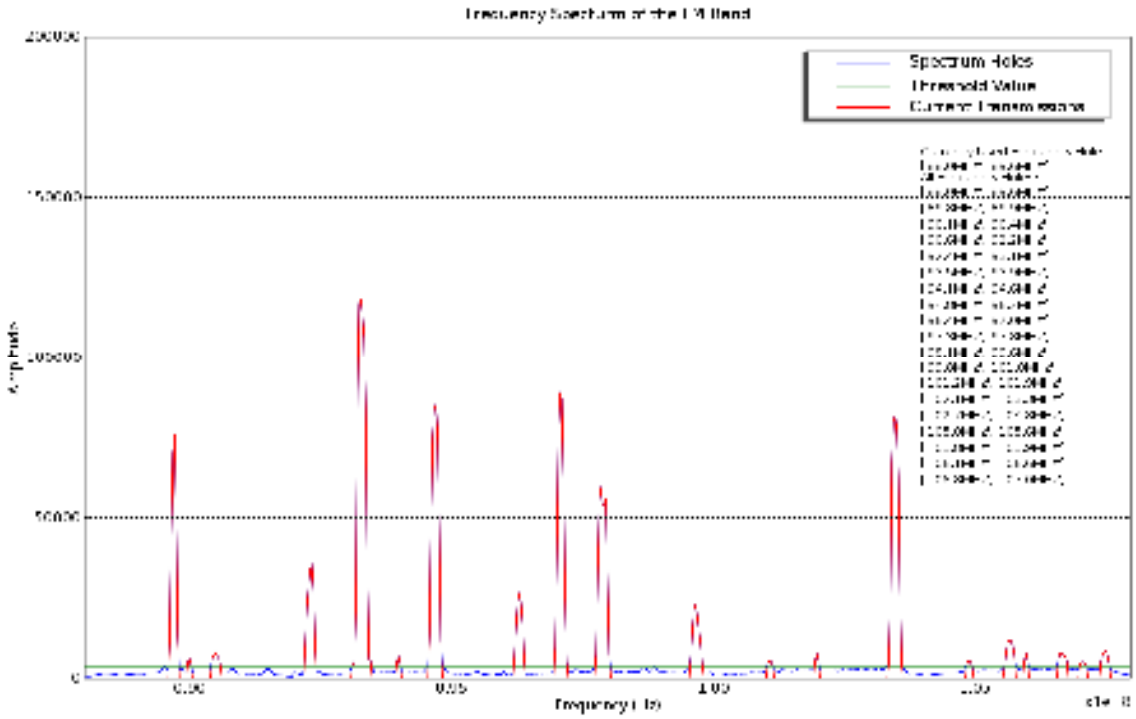

Figure 21 Spectrum Plot

The plot shows the transmissions as red and the spectrum holes as blue in the spectrum showing the user an overview of what is happening in the spectrum. This text on the plot shows the spectrum hole chosen to initiate the transmission. Since the data is recorded and plotted during the analysis process, it does not include the transmission initiated by the ISSR. In this case the spectrum hole chosen is 88.0MHz to 89.6MHz which can also be seen as a the very first blue section of the plot. This process is repeated over time as an infinite loop or with a maximum number of runs as specified by the user.

4.2 Performance Measures

As the experiment was performed, the level of performance was also observed. In case of ISSR, performance following performance measure was considered.

Percentage of Correct Spectrum Classification:

The main objective of the ISSR was to correctly analyze the spectrum and choose the frequency holes in the spectrum where no transmission is taking place. This was measured by taking a sample of 20 runs and verifying the frequency holes with a FM Radio. The following ratio was considered:

$$\frac{\text{Number of Correct Frequencies}}{\text{Number of total frequencies}} \times 100$$

$$\frac{\text{Number of Correct Frequencies}}{200} \times 100$$

After carrying out the 20 sample runs, 100% of the time, the frequencies were calculated correctly hence putting this performance measure result to a 100%.

4.3 Future Work

Intelligent Spectrum Sensor Radio is a constructive effort towards designing a fully functional cognitive radio. Even though all the aspects of cognitive radio are not covered by ISSR but it does give a good basic structure and only a little bit of future work can complete a cognitive radio system. A couple of future working options are a continuous transmission

instead of a current “bursty” transmission and detection of a new signal appearing in the frequency currently being used by the radio for its own transmission.

It is possible to make a continuous transmission. This could be done by adding a second USRP. The two USRP perform the same action as the ISSR designed in this thesis in an order of one after the other minimizing the delay between the transmissions to a virtually zero value.

Further improvement includes the detection of another transmission on the same frequency as the ISSR has already initiated the transmission. This can be done by calibrating the amplitudes of the self transmitted signal. As the amplitude increases a preset threshold, it can be recognized another transmission and self-transmission can be stopped immediately hence avoiding any interference with licensed users.

The main challenge remains of making the receiver of the system aware of what channel to use with respect to time. Therefore, a scheme needs to be worked on which can possibly involve a dedicated channel transmitting this information to the receiver. By doing this the transmitter can talk to the receiver that what sequence is being used at the transmission side and therefore it can use the same sequence to successfully retrieve information.

4.4 Conclusion

Completing the design and implementation of Intelligent Spectrum Sensor Radio (ISSR) is a great achievement which suggests that using GNU Radio and the USRP, implementation of cognitive radios is possible and is greatly simplified. This thesis project has come a long way in applying the theory of intelligent radios on to the hardware currently

available. It gives a good baseline structure to work on and ultimately will be able to come up with completely functional cognitive radio.

Bibliography

[1] Rudra, Angsuman. "Cognitive Radio: an Evolution From Software Radio." VME and Critical Systems. Dec. 2004. Feb. 2008 http://www.vmecritical.com/pdfs/ICS.Dec04.pdf

[2] "Software-Defined Radio: a Technology Overview." Broadcast Papers. Aug. 2002. Wipro Technologies. Feb. 2008 http://www.broadcastpapers.com/whitepapers/WiproSDRadio.pdf

[3] Blossom, Eric. "Software Radio." Comsec. Blossom Research, LLC. Feb. 2008 http://www.comsec.com/

[4] Shen, Dawei. "Tutorial 4: the USRP Board." Introduction. SDR Documentation. Notre Dame, IN: University of Notre Dame, 2005. Oct. 2007 http://www.nd.edu/~jnl/sdr/docs/

[5] Boyd, Susan, and Weber Shandwick. SDR Forum. Oftware Defined Radio Forum. Feb. 2008 http://www.sdrforum.org/

[6] Alberto. "SDRadio: a Software Defined Radio." The Weak Signal Pages of Alberto. Feb. 2008 http://www.sdradio.eu/

[7] Patton, Lee K. A GNU Radio Based Software-Defined Radar. Wright State University. 2007. 3-7

[8] Blossom, Eric. "Software Radio." Comsec. Blossom Research, LLC. Feb. 2008 http://www.comsec.com/

[9] Chen, Zhifeng, and Ke-Yu Chen. GNU Radio. Department of Electrical & Computer Engineering, University of Florida. 2006. Feb. 2008 http://www.wu.ece.ufl.edu/projects/softwareRadio/

[10] Python Programming Language -- Official Website. Dec. 2007 http://python.org/

[11] Blossom, Eric. "Exploring GNU Radio." Introduction. GNU Radio. Sept. 2007 http://www.gnu.org/software/gnuradio/doc/exploring-gnuradio.html

[12] Ettus, Matt. "USRP Datasheet." Ettus Research LLC. Oct. 2007 http://www.ettus.com/

[13] "USB 2.0, Hi Speed USB FAQ." Everything USB. Feb. 2008 http://www.everythingusb.com/

[14] Patton, Lee K. A GNU Radio Based Software-Defined Radar. Wright State University. 2007. 3-7

[15] "Mixed-Signal Front-End (MxFE™) Processor for Broadband Communications." Chart. http://www.analog.com/. Jan. 2008 http://www.analog.com/UploadedFiles/Data_Sheets/AD9860_9862.pdf

[16] Romano, Bruce. "FCC ADOPTS RULE CHANGES FOR SMART RADIOS." 10 Mar. 2005. Federal Communications Commission. Washington, D. C., 2005

[17] Merrill, Albert and Marsha Weiskopf. "Critical Issues in Spectrum Management for Defense Space Systems." Editorial. Crosslink 1 Nov. 2001

[18] "Fourier Analysis and FFT." Astro-Med Inc. Jan. 2008 http://www.astro-med.com/

[19] "FM Broadcast Band." Wikipedia. Feb. 2008 http://en.wikipedia.org/

[20] O'shea, Timothy J., T C. Clancy, and Hani J. Ebeid. PRACTICAL SIGNAL DETECTION AND CLASSIFICATION IN GNU RADIO. IEEE. 2007
[21] Reed, J H. Software Radio: a Modern Approach to Radio Engineering. Prentice Hall, 2002
[22] Isomäki, Petri, and Nastooh Avessta. An Overview of Software Defined. Turu Center for Computer Science, 2004. 11-16
[23] Unknown. Frequency Modulation. Feb. 2008 http://webtools.delmarlearning.com/
[24] "Getting Started." GNU Radio. Aug. 2007 http://gnuradio.org/
[25] Linux / Unix Command: Sox." About.Com. Feb. 2008 http://linux.about.com/
[26] Weisstein, Eric W. "Fourier Transform." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/FourierTransform.html, March 2008
[27] Smith, Julius O. Spectral Audio Signal Processing, March 2007 Draft, http://ccrma.stanford.edu/~jos/sasp/, online book, March 2008